4.Vorlesung Cluster-, Grid- und Cloud-Computing Hochschule Mannheim

Christian Baun, Viktor Mauch

Karlsruher Institut für Technologie
Steinbuch Centre for Computing
[baun|mauch]@kit.edu

1.4.2011

Wiederholung vom letzten Mal

- Was ist Cloud-Computing?
- Definitionen (lustige, falsche und richtige)
- Warum Cloud-Computing?
- Organisatorische Typen von Clouds
  - Public Cloud
  - Private Cloud
  - Hybrid Cloud
- Kategorien von Cloud-Systemen (anhand der angebotenen Dienste)
  - Humans as a Service (HuaaS)
  - Software as a Service (SaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (IaaS)
- Weitere Konzepte (Cloud Print, Cloud Gaming, Cloud Betriebssysteme)
- Fazit zum Cloud-Computing

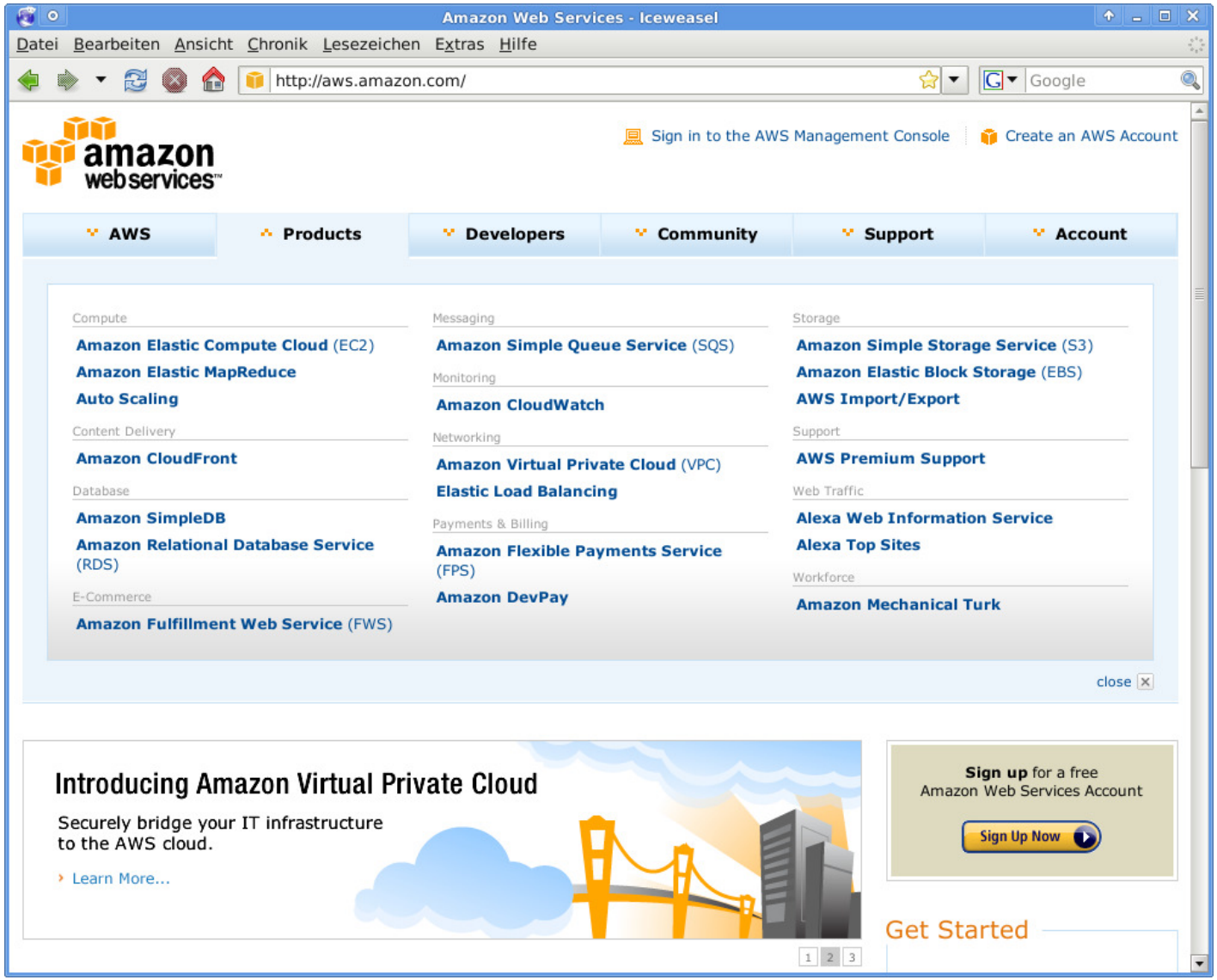
Heute

- Amazon Web Services (AWS)
  - Warum soll man die AWS einsetzen
  - Beispiele für Anwendungen, die AWS nutzen
  - Elastic Compute Cloud (EC2)
  - Elastic Block Store (EBS)
  - Elastic Load Balancing (ELB)
  - Simple Storage Service (S3)
  - Google Storage

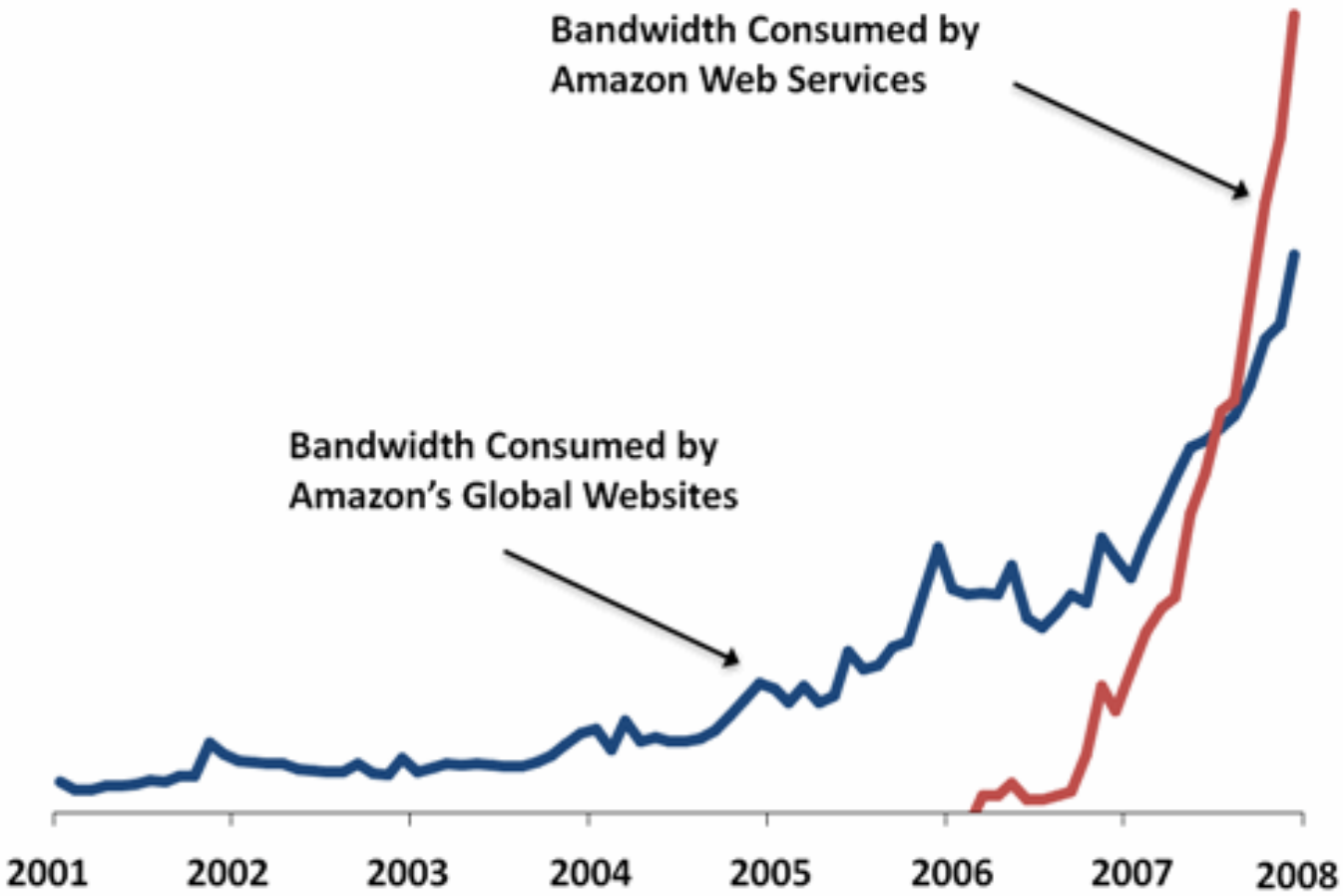
Amazon Web Services (AWS)

- Die AWS sind eine Sammlung verschiedener Cloud-Dienste
  - Gestartet bereits 2002
  - Abrechnung erfolgt nach Verbrauch
  - Bekannte Dienste innerhalb der AWS sind u.a. EC2, S3, EBS...
- Amazon Elastic Compute Cloud (EC2)
  - Dienst für virtuelle Server (Instanzen)
- Amazon Simple Storage Service (S3)
  - Dienst für Webobjekte
- Amazon Elastic Block Store (EBS)
  - Dienst für persistente Datenspeichervolumen
- Amazon SimpleDB
  - Verteiltes Datenbankmanagementsystem
- Amazon Simple Queue Service (SQS)
  - Dienst für Nachrichtenwarteschlangen (Message Queues)
- Amazon Mechanical Turk
  - Marktplatz für HuaaS/Crowdsourcing

AWS Übersicht – http://aws.amazon.com



Bandbreite der AWS (2008)



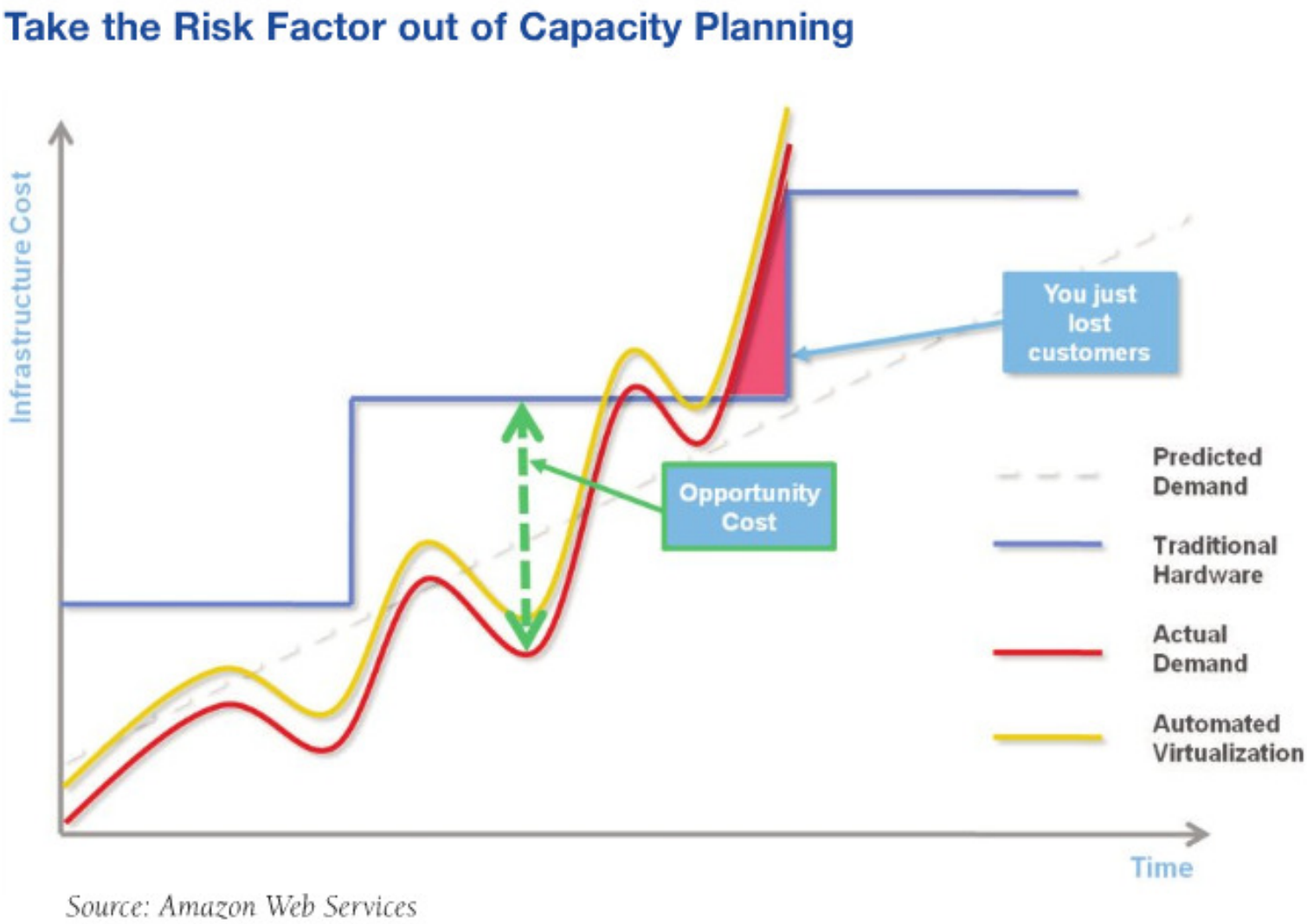
Quelle: `http://aws.typepad.com/aws/2008/05/lots-of-bits.html`

Warum AWS?

- Warum soll man auf die AWS zurückgreifen, anstatt eigene Resssourcen zu beschaffen?
- Welche Entwicklung des Ressourcenbedarf ist realistisch
- Szenario: Ein Web-Angebot eines jungen Unternehmens (Start-up)
  - Wie viele Ressourcen muss man vorhalten?
  - Wie verhält es sich mit den Kosten?
  - Wie viel Zeit ist notwendig, um zusätzliche Ressourcen zu beschaffen bzw. einzubinden?

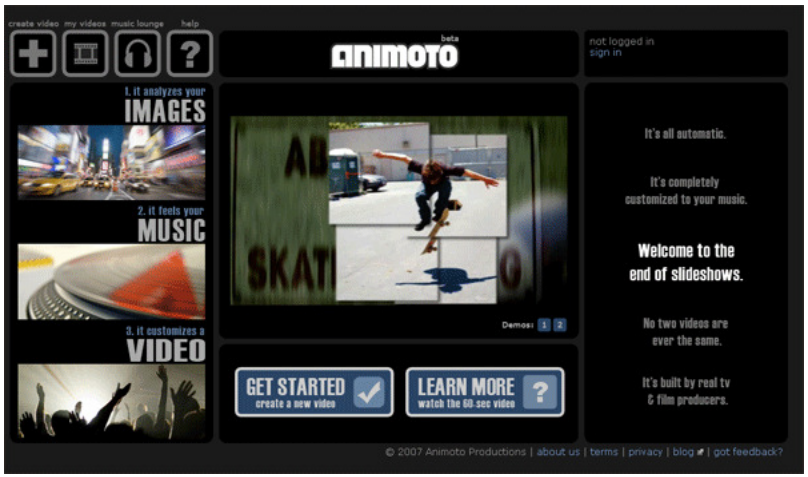
Ohne Kreditkarte geht bei den AWS nichts

Eigene Infrastruktur im Vergleich zur Cloud



Beispiel für AWS: Animoto – http://animoto.com

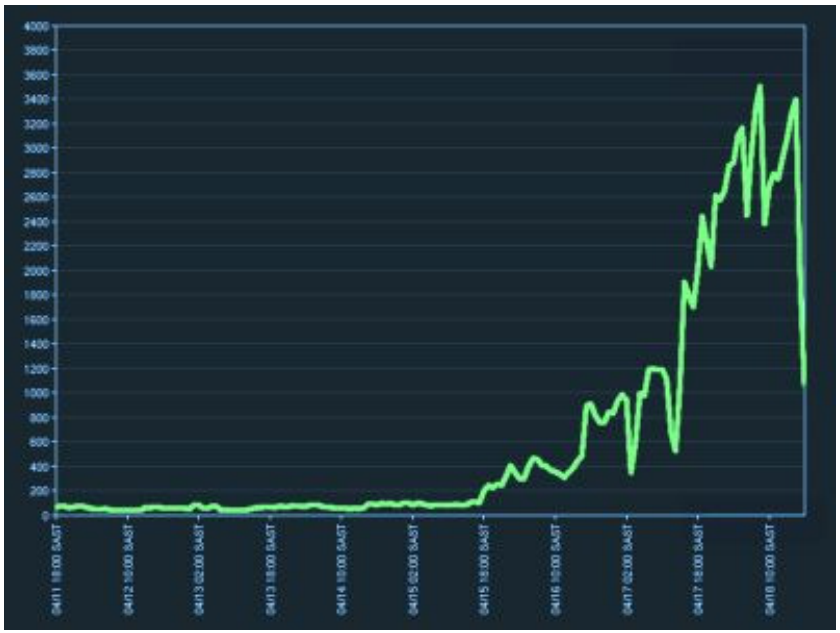
- Benutzer können Videos aus eigenen Bildern und Musik erstellen
- Die Bilder werden automatisch und passend mit der Musik unterlegt
- Die Software analysiert Bilder und Musik im generiert Videos im Stil eines Trailers bzw. MTV-Musikvideos
- Videos können direkt in YouTube eingestellt oder in diversen Formaten gespeichert werden



Quelle: http://profy.com/2007/08/15/animoto-easy-video

Beispiel für AWS: Animoto (2)

- Benutzerzahlen waren von 2006 bis 2008 konstant (niedrig)
- Im April 2008 auf Facebook erwähnt
  - 750.000 neue Benutzer in 3 Tagen
  - Bis zu 25.000 Benutzer wollten in einer Stunde ein Video erstellen
  - Slashdot-Effekt!
  - Automatische Anpassung der Instanzen zum Rendern der Videos von 2 auf bis zu 450



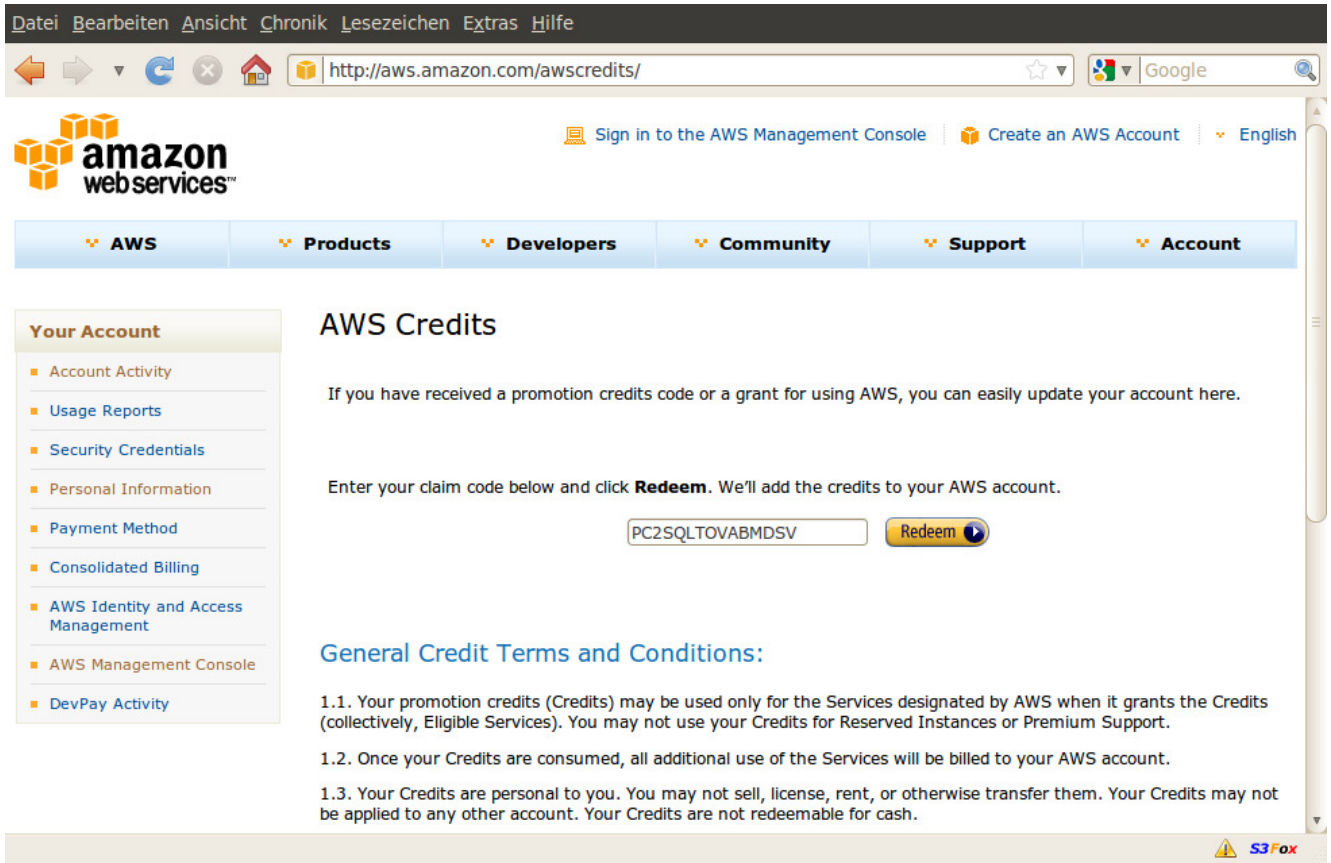
Slashdot-Effekt

- Lineare Steigerung der Benutzer-/Zugriffszahlen ist unrealistisch
- Großes Problem für Start-ups mit eigenen Ressourcen!

Beispiel für AWS: New York Times

- 2007: Die New York Times will PDF-Versionen der Artikel aus den Jahren 1851–1980 erstellen
  - Die Artikel 1851-1922 sollten komplett frei gegeben werden
- Die Artikel lagen in Form von 11 Millionen eingescannten Bildern vor
- Jeder Artikel musste aus verschiedenen TIFF-Dateien zusammengesetzt und skaliert werden
- 4 TB Daten wurden zuerst in S3 hochgeladen
- 100 EC2-Instanzen benötigten ca. 24 Stunden für die Berechnung
- Ergebnis: 1,5 TB PDF-Dateien in S3
- `http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/`
- `http://open.blogs.nytimes.com/2008/05/21/the-new-york-times-archives-amazon-web-services-timesmachine/`

AWS Credits – http://aws.amazon.com/awscredits



- Jeder Teilnehmer der Vorlesung bekommt eine AWS-Gutschrift
  - Wird mehr verbraucht, wird die Kreditkarte belastet!

Amazon Elastic Compute Cloud (EC2)

- Mit EC2 kann der Benutzer Instanzen virtueller Server auf in den Serverfarmen von Amazon starten, nutzen und steuern
  - Unterstützte Betriebssysteme: Linux, Windows Server 2003/2008 und OpenSolaris
- Virtuelle Server werden aus Amazon Machine Images (AMI) erzeugt
  - Blaupause für das Anlegen eines neuen virtuellen Servers
  - Amazon stellt vorgefertigte Images zur Verfügung
  - Auch Drittanbieter wie z.B. IBM, Oracle und Sun stellen AMIs mit eigenen Software-Paketen zur Verfügung
  - Auch die Benutzer können eigene Images anfertigen
  - Es ist möglich, eigene AMIs zu veröffentlichen und über eine Produkt-ID zu vermarkten (Paid Instances)

Von EC2 unterstützte Betriebssysteme

- Red Hat Enterprise Linux
- openSUSE Linux
- Ubuntu Linux
- Debian
- Fedora
- Gentoo Linux
- Oracle Enterprise Linux
- OpenSolaris
- Windows Server 2003/2008

EC2 Begrifflichkeiten

- EC2 hat aktuell 5 Standorte, in denen sich Ressourcen befinden
  `US East` $\Longrightarrow$ Virginia
  `US West` $\Longrightarrow$ Kalifornien
  `EU East` $\Longrightarrow$ Irland
  `Asia Pacific` (Southeast) $\Longrightarrow$ Singapur
  `Asia Pacific` (Northeast) $\Longrightarrow$ Tokio
- Die Standorte werden als **Regionen** bezeichnet
- Jeder Standort enthält Verfügbarkeitszonen (**Availability Zones**)
- Jede Verfügbarkeitszone ist ein in sich abgeschlossener Cluster
  `us-east-1a`, `us-east-1b`, `us-east-1c`, `us-east-1d`
  `us-west-1a`, `us-west-1b`
  `eu-west-1a`, `eu-west-1b`
  `ap-southeast-1a`, `ap-southeast-1b`
  `ap-northeast-1a`

EC2 Instanztypen

Instanztyp	Architektur	virt. Cores	ECU	RAM	I/O Perf.
`t1.micro`	32 + 64 Bit	1	max. 2	613 MB	Low
`m1.small`	32 Bit	1	1	1,7 GB	Moderate
`c1.medium`	32 Bit	2	5	1,7 GB	Moderate
`m1.large`	64 Bit	2	4	7,5 GB	High
`m1.xlarge`	64 Bit	4	8	15 GB	High
`m2.xlarge`	64 Bit	2	6,5	17,1 GB	Moderate
`m2.2xlarge`	64 Bit	4	13	34,2 GB	High
`m2.4xlarge`	64 Bit	8	26	68,4 GB	High
`c1.xlarge`	64 Bit	8	20	7 GB	High
`cc1.4xlarge`	64 Bit	8	33,5	23 GB	Very High
`cg1.4xlarge`	64 Bit	8	33,5	22 GB	Very High

- 1 ECU (EC2 Compute Unit) ist äquivalent zu einer 2007er AMD Opteron oder Intel Xeon CPU mit 1,0 bis 1,2 GHz

EC2 – Der normale Ablauf (1)

- Benutzer braucht ein Schlüsselpaar für die Identifikation gegenüber seiner Instanzen
  - Einloggen ohne Passwort (Public-Key Verfahren)
  - Öffentliche Schlüssel wird in den Instanzen hinterlegt
  - Private Schlüssel bleibt auf der Arbeitsumgebung des Benutzers
  - Ein neues Schlüsselpaar kann angelegt oder auf ein bestehendes Schlüsselpaar zurückgegriffen werden
- Benutzer überlegt sich, welche Ports offen sein müssen
  - Je weniger Ports offen sind, desto besser (sicherer)
  - Benutzer legt eine Security Group für die neue Instanz an, in der die gewünschten Ports freigegeben sind
  - Eventuell kann auf eine bestehende Security Group zurückgegriffen werden
- Benutzer überlegt sich, welches Betriebssystem (AMI) und welcher Instanztyp passend für die eigenen Anforderungen ist
- Instanz starten mit den Parametern aus den obigen Vorüberlegungen

EC2 – Der normale Ablauf (2)

- Nach dem Startprozess erhält der entstandene virtuelle Server eine dynamisch zugewiesene öffentliche und eine private IP-Adresse
  - Unter der öffentlichen Adresse ist der Rechner aus dem Internet erreichbar
  - Unter der privaten Adresse ist er für andere Instanzen in der Amazon Cloud sichtbar
- Private und öffentliche Adressen werden dynamisch bei jedem Start einer Instanz neu vergeben
  - Daher ungeeignet für den dauerhaften Betrieb eines Servers
  - Server müssen immer mal wieder neu gestartet werden
  - Lösung: Elastic IPs
- Benutzer können Elastic IPs, die – einmal reserviert – immer wieder den eigenen Servern zuweisen

Persistenz der Daten bei EC2

- Bei der Beendigung einer Instanz gehen alle Änderungen verloren
- Wertvolle Daten müssen außerhalb der Instanz gespeichert werden
  - Speicherung großer Mengen schwach strukturierter Daten kann in S3 erfolgen
  - Block-basierten Zugriff auf ein Speichermedium bietet EBS

Kosten von EC2: On-Demand Instances

US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

Standard On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Small (Default)	$0.085 per hour	$0.12 per hour
Large	$0.34 per hour	$0.48 per hour
Extra Large	$0.68 per hour	$0.96 per hour
Micro On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Micro	$0.02 per hour	$0.03 per hour
High-Memory On-Demand Instances		
Extra Large	$0.50 per hour	$0.62 per hour
Double Extra Large	$1.00 per hour	$1.24 per hour
Quadruple Extra Large	$2.00 per hour	$2.48 per hour
High-CPU On-Demand Instances		
Medium	$0.17 per hour	$0.29 per hour
Extra Large	$0.68 per hour	$1.16 per hour
Cluster Compute Instances		
Quadruple Extra Large	$1.60 per hour	N/A*

* Windows is not currently available for Cluster Compute Instances.

Kosten von EC2: Reserved Instances

US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

	One-time Fee			
Standard Reserved Instances	**1 yr Term**	**3 yr Term**	**Linux/UNIX Usage**	**Windows Usage**
Small (Default)	$227.50	$350	$0.03 per hour	$0.05 per hour
Large	$910	$1400	$0.12 per hour	$0.20 per hour
Extra Large	$1820	$2800	$0.24 per hour	$0.40 per hour
Micro Reserved Instances				
Micro	$54	$82	$0.007 per hour	$0.013 per hour
High-Memory Reserved Instances				
Extra Large	$1325	$2000	$0.17 per hour	$0.24 per hour
Double Extra Large	$2650	$4000	$0.34 per hour	$0.48 per hour
Quadruple Extra Large	$5300	$8000	$0.68 per hour	$0.96 per hour
High-CPU Reserved Instances				
Medium	$455	$700	$0.06 per hour	$0.125 per hour
Extra Large	$1820	$2800	$0.24 per hour	$0.50 per hour
Cluster Compute Reserved Instances				
Quadruple Extra Large	$4290	$6590	$0.56 per hour	N/A*

* Windows is not currently available for Cluster Compute Instances.

Kosten von EC2: Data Transfer

Internet Data Transfer

The pricing below is based on data transferred "in" and "out" of Amazon EC2.

Data Transfer In	**US & EU Regions**	**APAC Region**
All Data Transfer	Free until Nov 1, 2010 *	Free until Nov 1, 2010 *

Data Transfer Out **	**US & EU Regions**	**APAC Region**
First 1 GB per Month	$0.00 per GB	$0.00 per GB
Up to 10 TB per Month	$0.15 per GB	$0.19 per GB
Next 40 TB per Month	$0.11 per GB	$0.15 per GB
Next 100 TB per Month	$0.09 per GB	$0.13 per GB
Over 150 TB per Month	$0.08 per GB	$0.12 per GB

Kosten von EC2: Elastic IP und Load Balancing

Elastic IP Addresses

No cost for Elastic IP addresses while in use

- $0.01 per non-attached Elastic IP address per complete hour
- $0.00 per Elastic IP address remap – first 100 remaps / month
- $0.10 per Elastic IP address remap – additional remap / month over 100

Elastic Load Balancing

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore

- $0.025 per Elastic Load Balancer-hour (or partial hour)
- $0.008 per GB of data processed by an Elastic Load Balancer

EC2 Software Appliances

Software

Amazon EC2 enables our partners and customers to build and customize Amazon Machine Images (AMIs) with software based on your needs. We have hundreds of free and paid AMIs available for you to use. A small sampling of the software available for use today within Amazon EC2 includes:

Databases	**Batch Processing**	**Web Hosting**
IBM DB2	Hadoop	Apache HTTP
IBM Informix Dynamic Server	Condor	IIS/Asp.Net
Microsoft SQL Server Standard 2005	Open MPI	IBM Lotus Web Content Management
MySQL Enterprise		IBM WebSphere Portal Server
Oracle Database 11g		

Application Development Environments	**Application Servers**	**Video Encoding & Streaming**
IBM sMash	IBM WebSphere Application Server	Wowza Media Server Pro
JBoss Enterprise Application Platform	Java Application Server	Windows Media Server
Ruby on Rails	Oracle WebLogic Server	

Verfügbarkeit von EC2

- Amazon garantiert für EC2 eine jährliche Verfügbarkeit von 99.95%
- Maximale Downtime: ca. 263 Minuten (ca. 4,5 Stunden) pro Jahr
- Wenn die Verfügbarkeit unterschritten wird, bekommt der Benutzer eine Gutschrift von 10%
  - Aber was bringt eine Gutschrift über 10%, wenn die eigenen Dienste nicht mehr erreichbar sind?
- `http://aws.amazon.com/ec2-sla/`

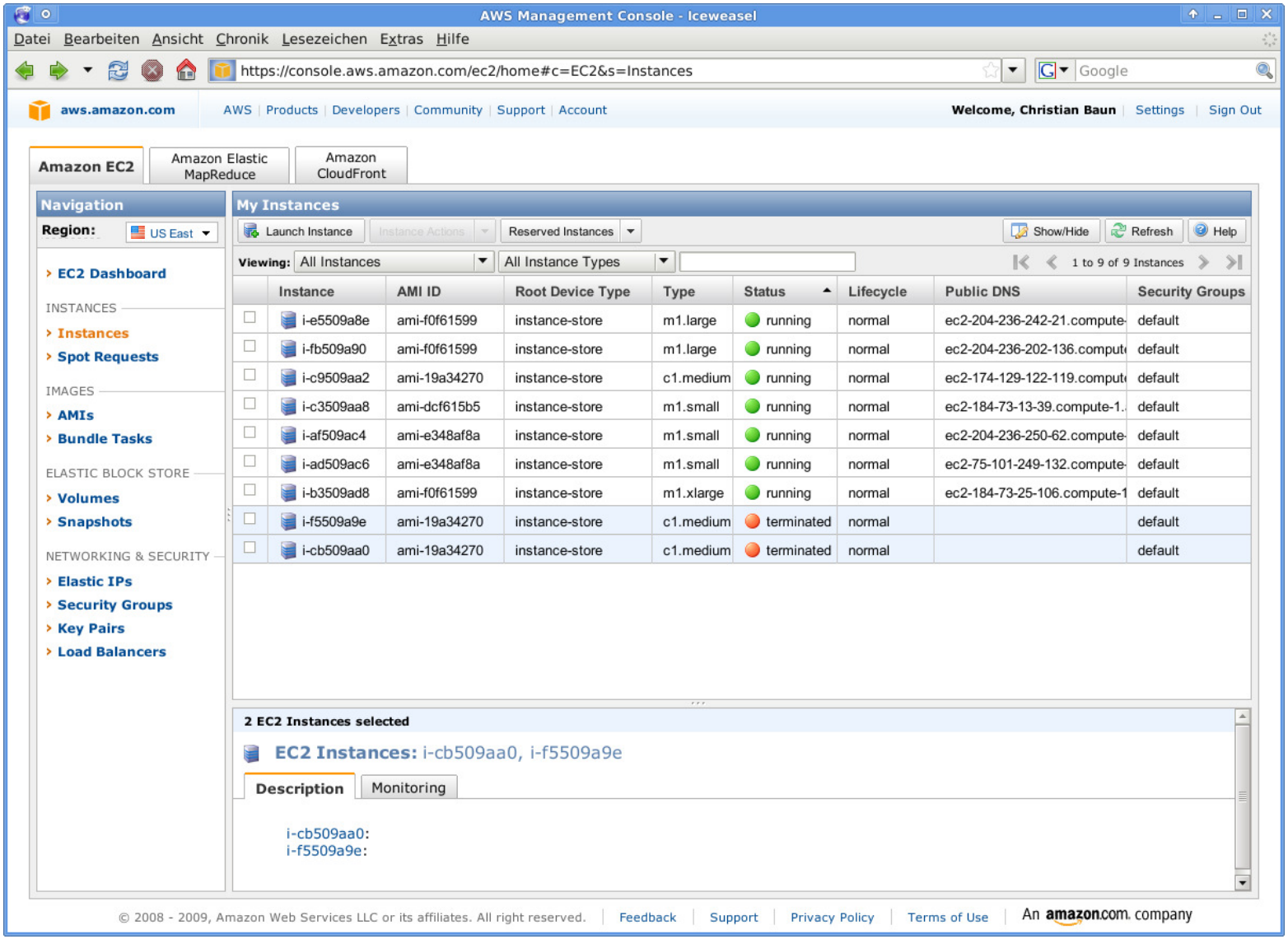
Service Commitments and Service Credits

If the Annual Uptime Percentage for a customer drops below 99.95% for the Service Year, that customer is eligible to receive a Service Credit equal to 10% of their bill (excluding one-time payments made for Reserved Instances) for the Eligible Credit Period. To file a claim, a customer does not have to have wait 365 days from the day they started using the service or 365 days from their last successful claim. A customer can file a claim any time their Annual Uptime Percentage over the trailing 365 days drops below 99.95%.

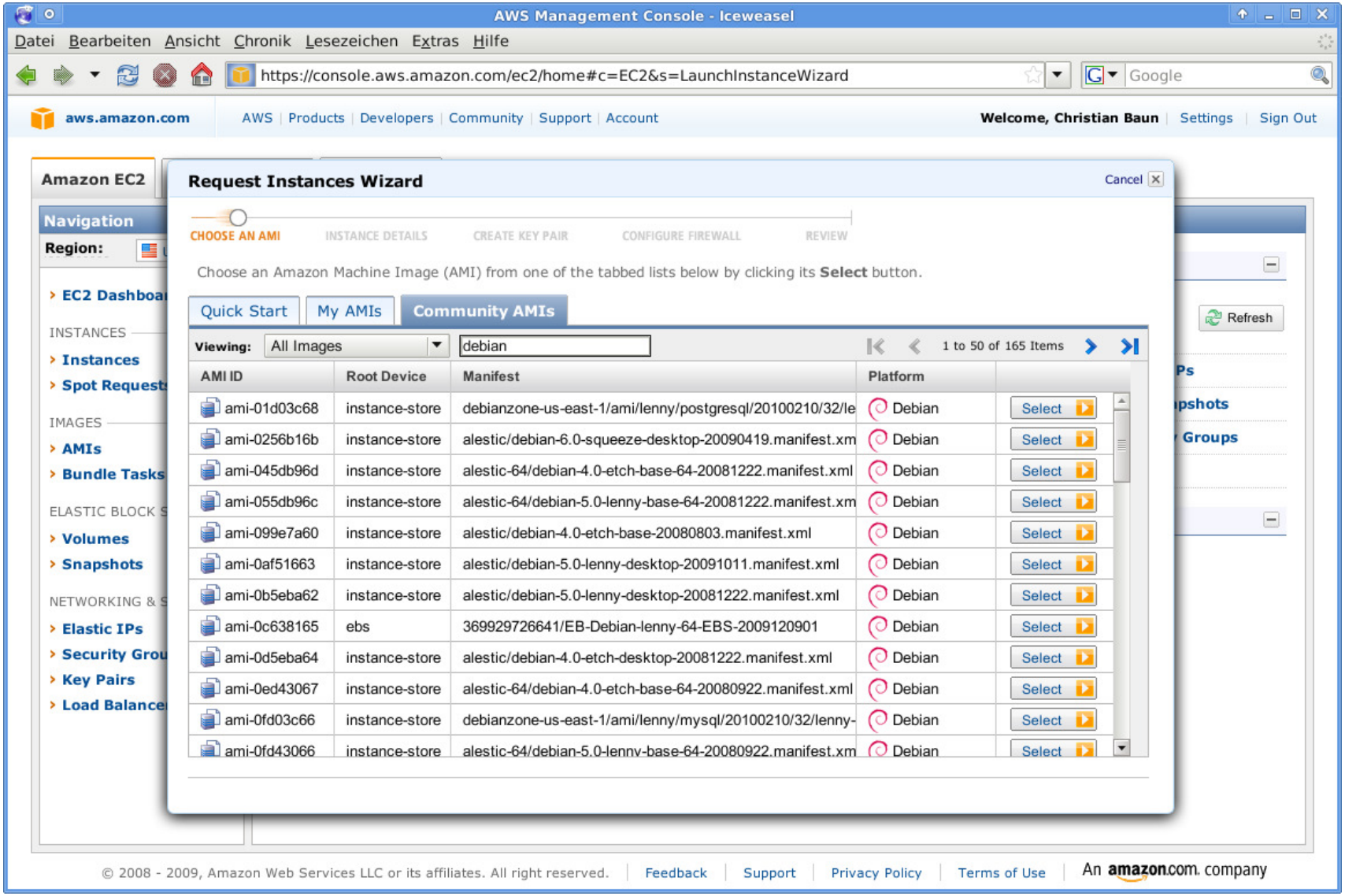
Mit EC2 arbeiten

- Kommandozeilenprogramme
  - Amazon EC2 Getting Started Guide
    `http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/`
  - Amazon EC2 Quick Reference Card
    `http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-qrc.pdf`
- Firefox-Plugins (Elasticfox und Hybridfox)
  - `http://s3.amazonaws.com/ec2-downloads/elasticfox.xpi`
  - `http://code.google.com/p/hybridfox/`
- Web-Anwendungen/SaaS (AWS Management Console, Ylastic, KOALA)
  - `http://aws.amazon.com/console/`
  - `http://ylastic.com`
  - `http://koalacloud.appspot.com`

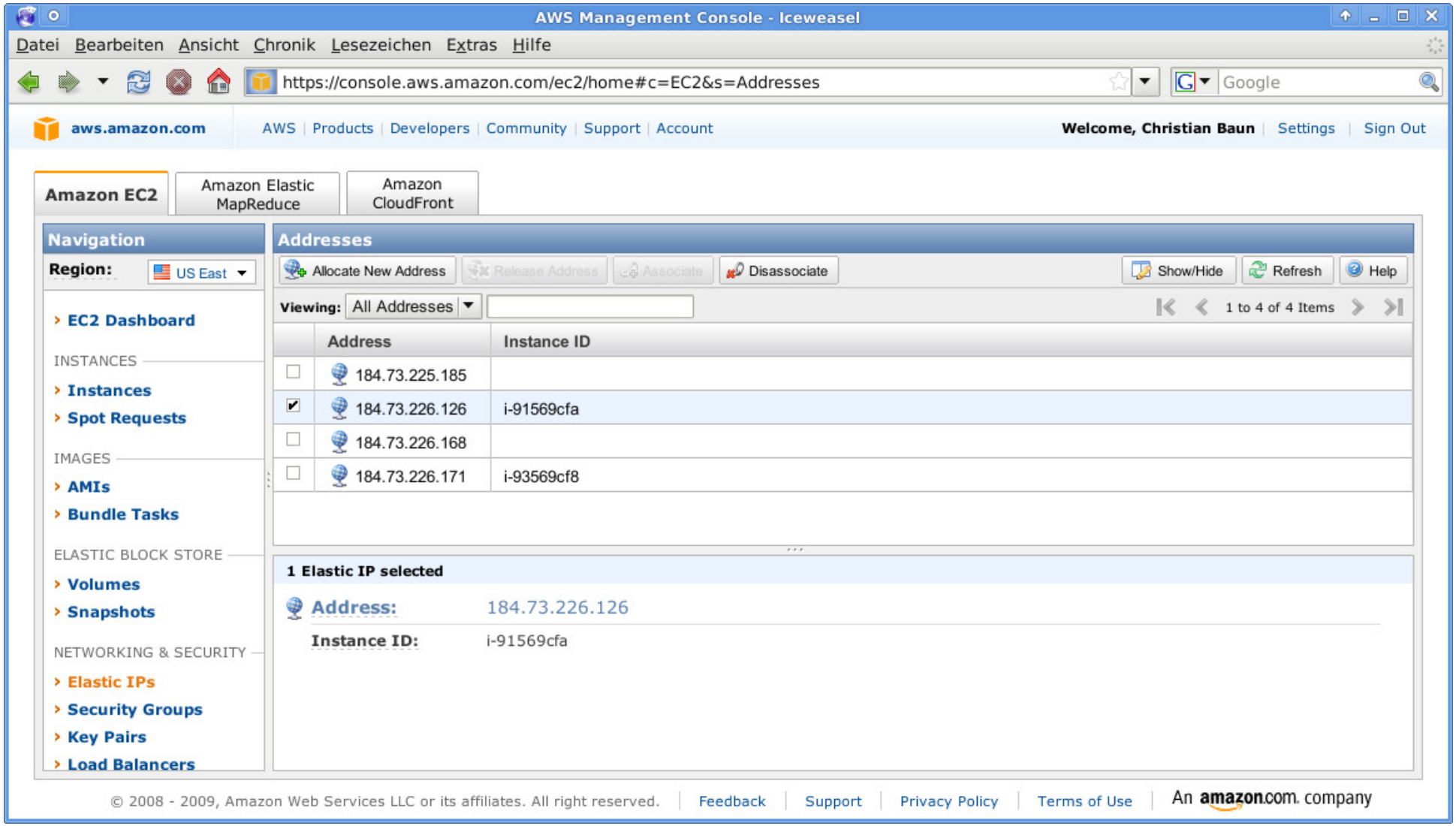
AWS Management Console (Instanzen)



AWS Management Console (AMIs)

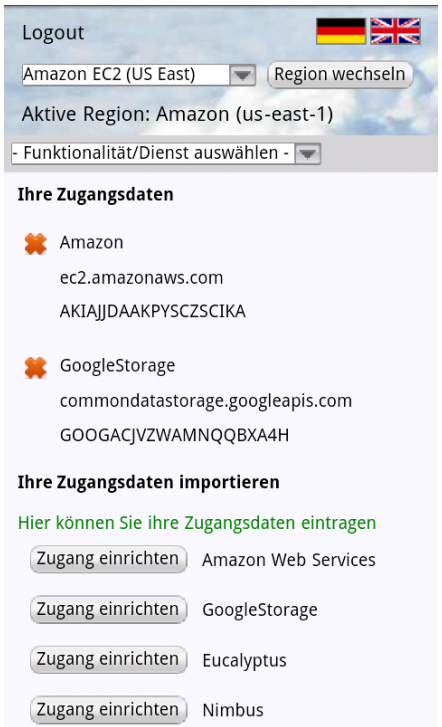


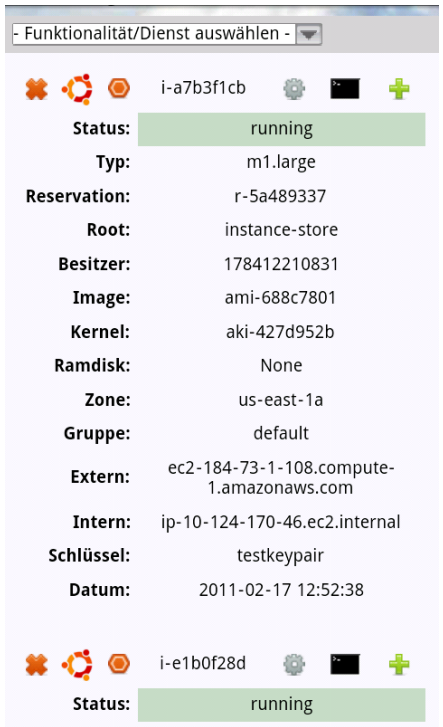
AWS Management Console (Elastic IPs)



KOALA

- **K**arlsruhe **O**pen **A**pplication for c**L**oud **A**dministration
  - `http://koalacloud.appspot.com`
- Webanwendung die die Arbeit mit AWS-kompatiblen Infrastruktur- und Speicherdiensten unterstützt





Mit der EC2 API und boto arbeiten

- Einfacher Zugriff auf die EC2 API mit boto und Python
  `http://code.google.com/p/boto/`

```
#!/usr/bin/env python

from boto.ec2.connection import EC2Connection
# Verbindung zu EC2 aufbauen
# Variable "conn" zeigt auf ein "EC2Connection" Objekt
conn = EC2Connection('<aws access key>', '<aws secret key>')

# Liste aller Regionen erhalten und ausgeben
liste_regionen = conn.get_all_regions()
print liste_regionen

# Liste aller Zonen erhalten und ausgeben
liste_zonen = conn.get_all_zones()
print liste_zonen
```

Mit der EC2 API und boto arbeiten (Fortsetzung)

```
# Liste aller Sicherheitsgruppen erhalten und ausgeben
liste_gruppen = conn.get_all_security_groups()
print liste_gruppen

# Liste aller Schluesselpaare erhalten und ausgeben
liste_keys = conn.get_all_key_pairs()
print liste_keys

# Instanz(en) starten
reservation = conn.run_instances('ami-e348af8a',
                                 min_count=2,
                                 key_name='geheim',
                                 instance_type='m1.small')

# Liste aller Instanzen erhalten und ausgeben
liste_instanzen = conn.get_all_instances()
print liste_instanzen
```

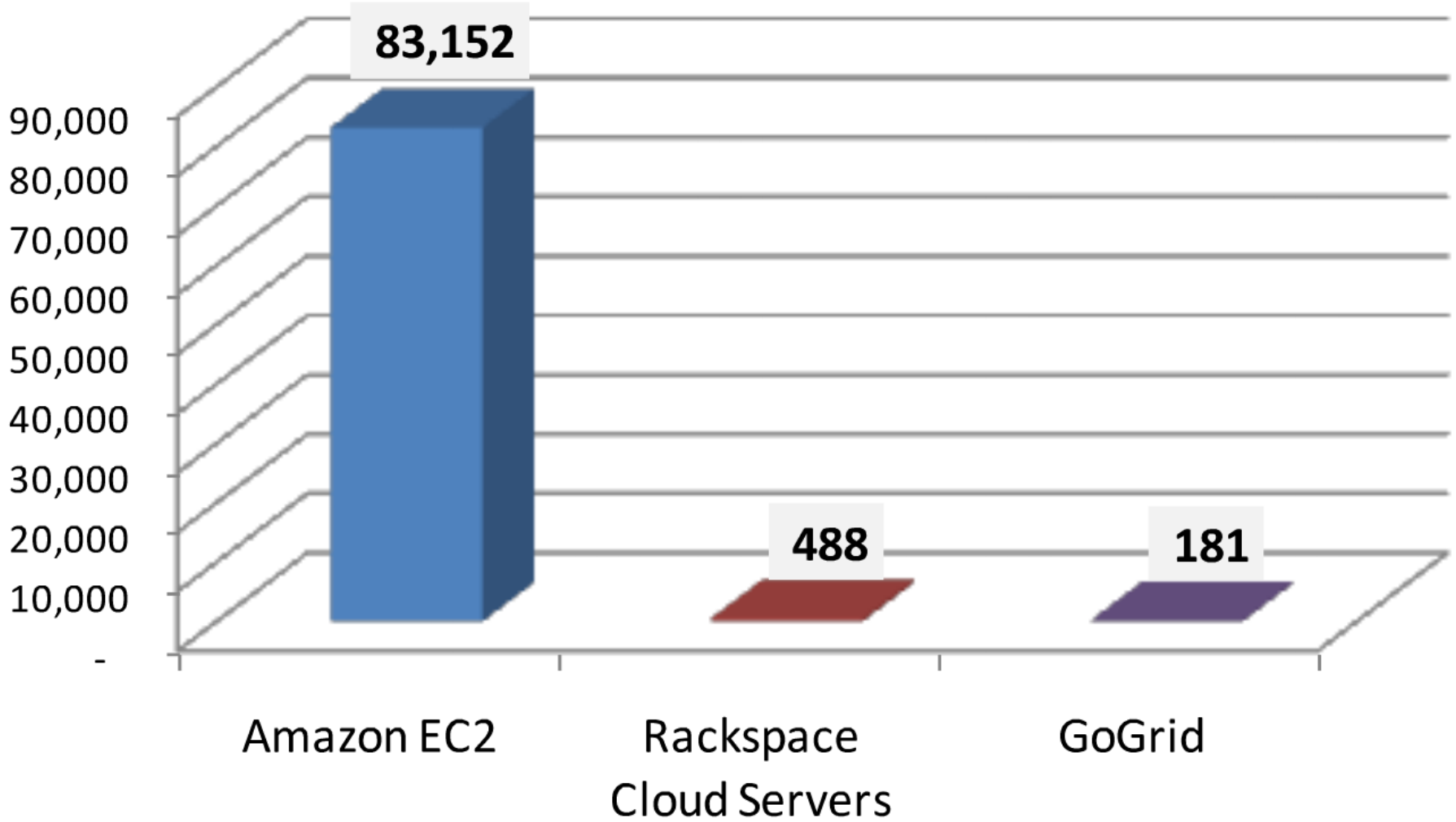
EC2 IDs

Naheliegende Fragen

Wie viele Instanzen (Server) werden jeden Tag in der Cloud gestartet und wie viele Instanzen wurden überhaupt schon in der Cloud gestartet?

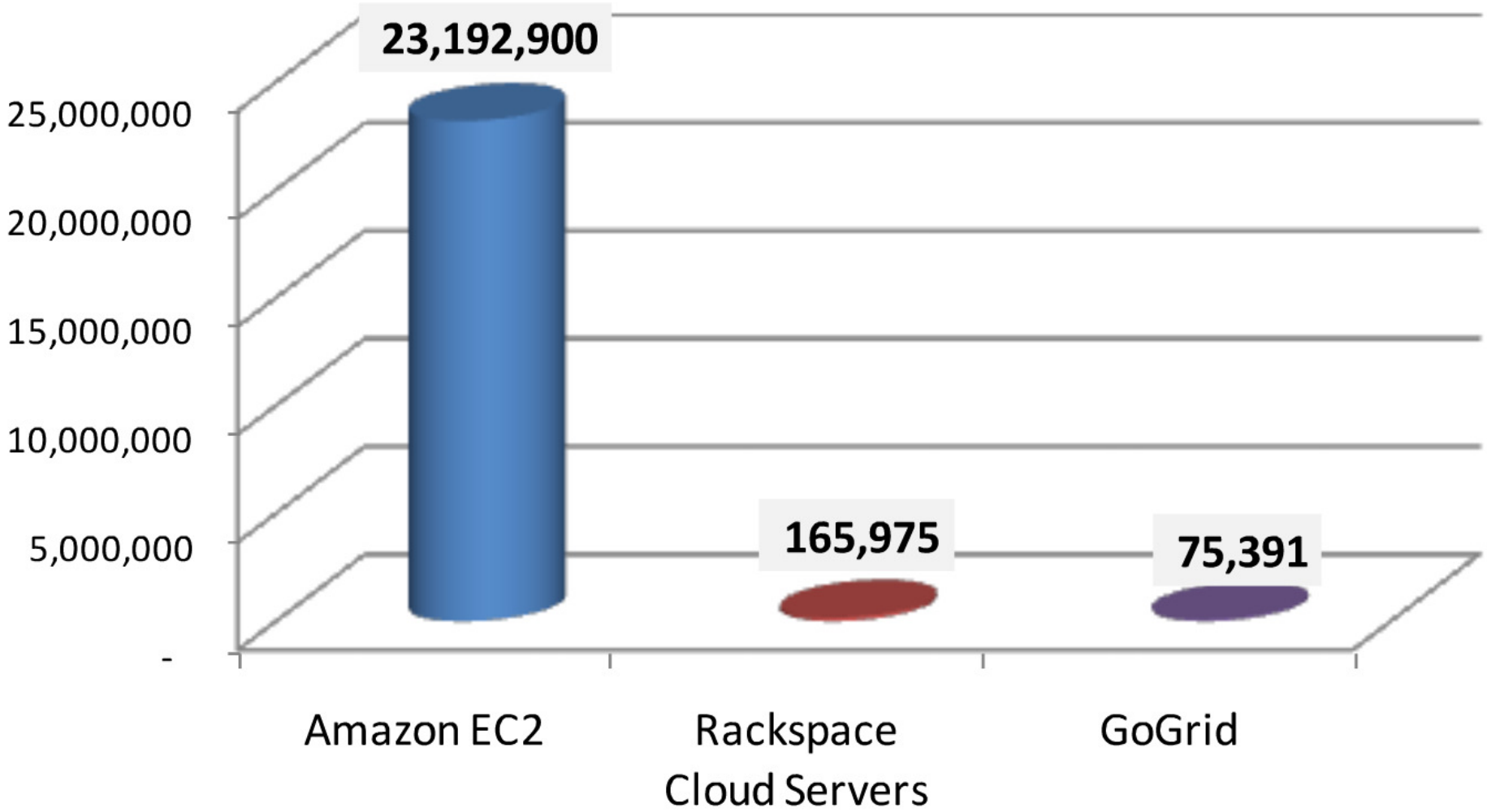
- Problem:
  - Bei EC2 sind die IDs der Instanzen auf den ersten Blick nicht sehr aussagekräftig
  - Beispiele:
    `i-60da3d07`
    `i-d334d41c`
    `i-4167f92a`
    `i-27e7004b`
    `i-45e7008a`
- Guy Rosen hat die Zusammensetzung der Instanz-IDs bei EC2 herausgefunden
  `http://www.jackofallclouds.com`

EC2: Instanzen pro Tag



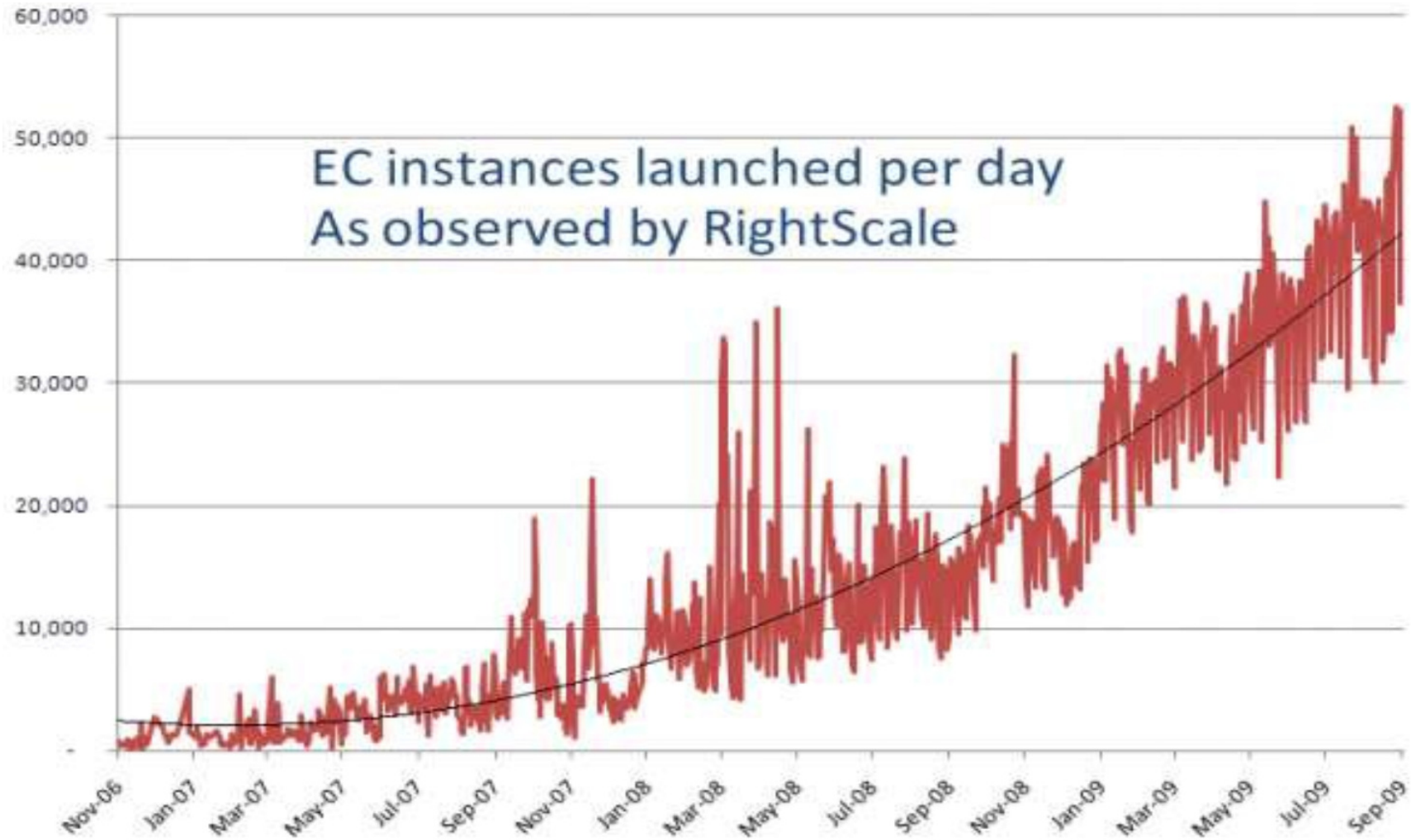
- Quelle: Guy Rosen

EC2: Instanzen insgesamt



- Quelle: Guy Rosen

EC2: Instanzen pro Tag (Entwicklung)



- Quelle: Guy Rosen

Amazon Elastic Block Store (EBS)

- EBS ist ein Teil von EC2
- Innerhalb jeder Verfügbarkeitszone können Benutzer EBS-Volumes erzeugen
  - Größe: 1 GB bis 1 TB
- Ein EBS-Volume ist persistenter Speicher
- Ein neues Volume ist wie ein unformatiertes Block-Gerät
- Ein Volume kann immer an genau eine Instanz angehängt werden
  - Dafür müssen sich Volume und Instanz zu innerhalb der gleichen Verfügbarkeitszone befinden
- Ein Volume kann ein beliebiges Dateisystem enthalten
- Ein Volume ist von der Bedienung her exakt wie ein USB-Stick
- Merke: EBS ist Speicher für Menschen und S3 für Anwendungen
- Volume-Snapshots können erzeugt (und in S3 gespeichert) werden

Kosten von EBS

Amazon Elastic Block Store

US – N. Virginia | US – N. California | EU – Ireland

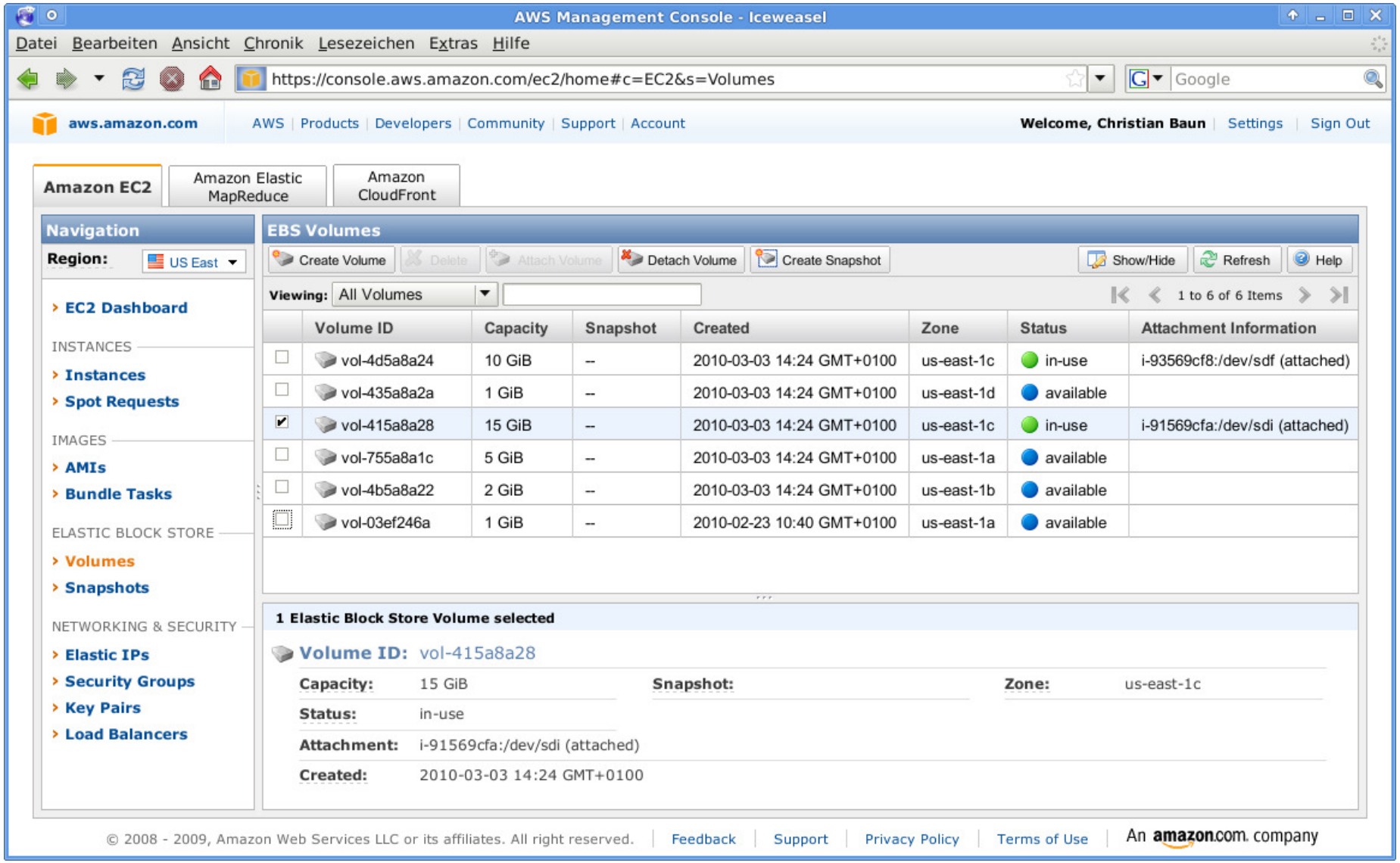
Amazon EBS Volumes

- $0.10 per GB-month of provisioned storage
- $0.10 per 1 million I/O requests

Amazon EBS Snapshots to Amazon S3 (priced the same as Amazon S3)

- $0.15 per GB-month of data stored
- $0.01 per 1,000 PUT requests (when saving a snapshot)
- $0.01 per 10,000 GET requests (when loading a snapshot)

AWS Management Console (EBS-Volumes)



Mit der EBS und boto arbeiten

```
#!/usr/bin/env python

from boto.ec2.connection import EC2Connection
# Verbindung zu EC2 aufbauen
# Variable "conn" zeigt auf ein "EC2Connection" Objekt
conn = EC2Connection('<aws access key>', '<aws secret key>')

# Volume (1 GB) in Region "us-east-1a" erzeugen und ID ausgeben
volume = conn.create_volume(1, 'us-east-1a')
print volume.id

# Volume "vol-1e0f0677" entfernen
conn.delete_volume('vol-1e0f0677')
```

- Volume an Instanz anhängen $\Longrightarrow$ `attach_volume()`
- Volume von Instanz lösen $\Longrightarrow$ `detach_volume()`

Amazon Elastic Load Balancing (ELB)

- ELB ist ein Teil von EC2
- Innerhalb jeder Verfügbarkeitszone können Benutzer elastische Lastverteiler erzeugen
- Jedem Lastverteiler weist der Benutzer einen Pool an Instanzen zu
- Ein elastischer Lastverteiler verteilt automatisch die eingehenden Anfragen über die EC2-Instanzen seines Pools
- Ein ELB ermittelt fehlerhafte Instanzen innerhalb seines Pools und leitet die Anfragen automatisch an fehlerfreie Instanzen im Pool weiter

ELB anlegen

Create a New Load Balancer Cancel

DEFINE LOAD BALANCER — CONFIGURE HEALTH CHECK — ADD EC2 INSTANCES — REVIEW

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80. We also provide several application examples to assist you in opening up the right ports.

Load Balancer Name: TestLoadBalancer

Listener Configuration:

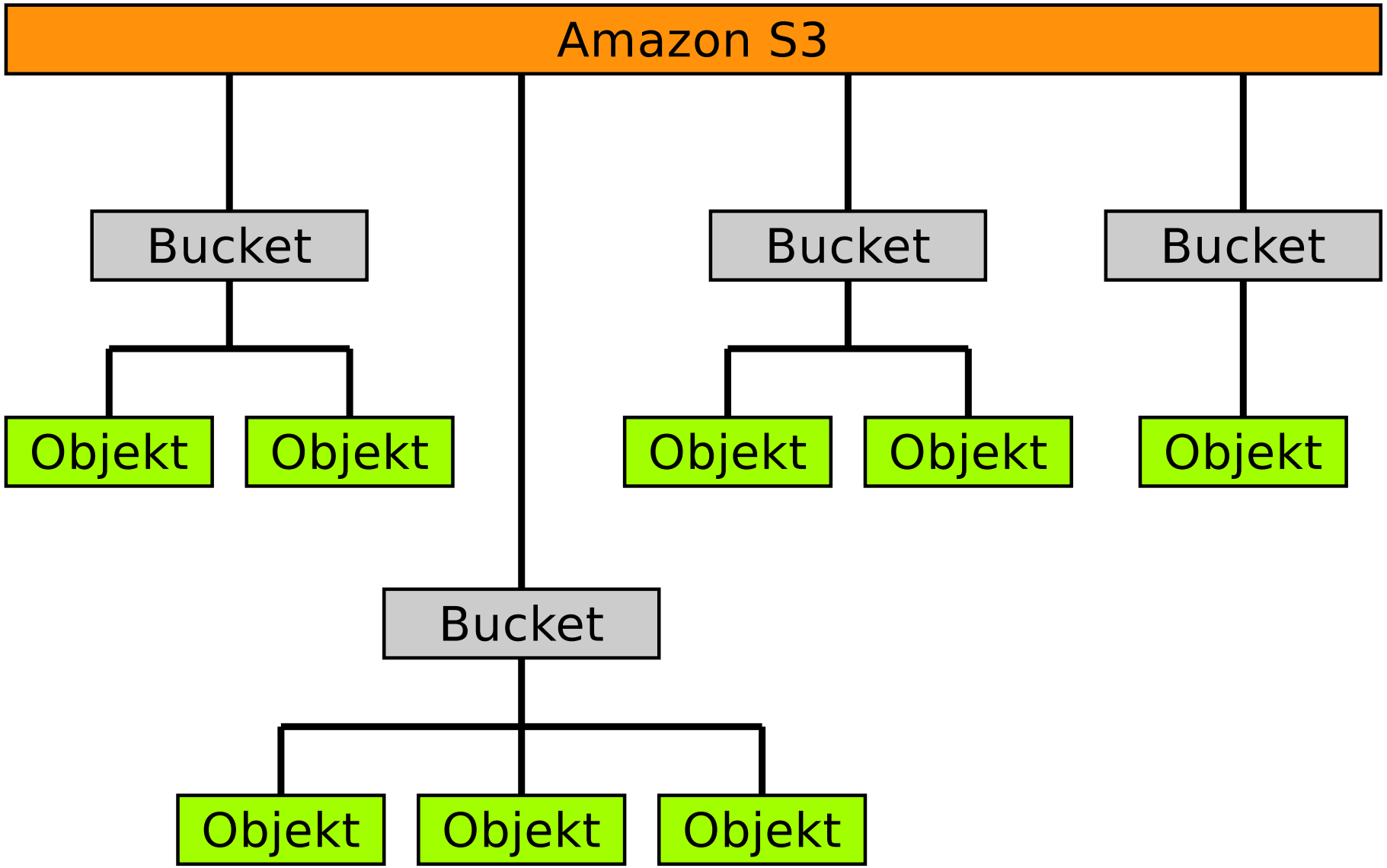
Common Applications	Protocol	Load Balancer Port	EC2 Instance Port	Actions
Apache HTTP Server	HTTP	80	80	Remove
Secure HTTP Server	HTTPS (Secure HTTP)	443	80	Remove
Custom...	TCP	21	21	Save

Continue

Amazon Simple Storage Service (S3)

- In S3 werden Daten in Form von Webobjekten gespeichert
- In S3 existieren keine Dateien oder Verzeichnisse, sondern nur Keys
  - Jeder Key kann maximal 5 GB groß sein
  - Zu jedem Key werden 2 KB Metadaten gespeichert
- Jeder Key muss einem Bucket zugeordnet sein (sich innerhalb eines Buckets befinden)
  - Jeder Bucket innerhalb von S3 muss einen eindeutigen Namen haben
  - Ein Bucket kann keinen weiteren Bucket enthalten
- Für jeden Key und Bucket können die Zugriffsreche festgelegt werden
  - Access Control List (ACL)
  - `http://s3.amazonaws.com/bucket/key`
  - `http://bucket.s3.amazonaws.com/key`
- Zufriff auf Buckets and Keys geschieht via REST oder SOAP
- Keys können auch via BitTorrent heruntergeladen werden (muss man freigeben)

Namensraum von S3



Entwicklung der S3 Nutzung

Zeitraum	**Gespeicherte Objekte**
August 2006	800 Millionen
April 2007	5 Milliarden
Oktober 2007	10 Milliarden
Januar 2008	14 Milliarden
Oktober 2008	29 Milliarden
März 2009	52 Milliarden
August 2009	64 Milliarden
Oktober 2009	82 Milliarden
März 2010	102 Milliarden
Oktober 2010	???

Kosten von S3

US – Standard | US – N. California | EU – Ireland | APAC – Singapore

Storage (Designed for 99.999999999% Durability)		Reduced Redundancy Storage (Designed for 99.99% Durability)		Data Transfer*		Requests	
Tier	**Pricing**	**Tier**	**Pricing**	**Tier**	**Pricing**	**Type**	**Pricing**
First 50 TB / Month of Storage Used	$0.150 per GB	First 50 TB / Month of Storage Used	$0.100 per GB	All Data Transfer In	Free until November 1st, 2010**	PUT, COPY, POST, or LIST	$0.01 per 1,000 Requests
Next 50 TB / Month of Storage Used	$0.140 per GB	Next 50 TB / Month of Storage Used	$0.093 per GB	First 1 GB / month data transfer out	$0.000 per GB	GET and All Other Requests***	$0.01 per 10,000 Requests
Next 400 TB / Month of Storage Used	$0.130 per GB	Next 400 TB / Month of Storage Used	$0.087 per GB	Up to 10 TB / month data transfer out	$0.150 per GB		
Next 500 TB / Month of Storage Used	$0.105 per GB	Next 500 TB / Month of Storage Used	$0.070 per GB	Next 40 TB / month data transfer out	$0.110 per GB		
Next 4000 TB / Month of Storage Used	$0.080 per GB	Next 4000 TB / Month of Storage Used	$0.053 per GB	Next 100 TB / month data transfer out	$0.090 per GB		
Storage Used / Month Over 5000 TB	$0.055 per GB	Storage Used / Month Over 5000 TB	$0.037 per GB	Greater than 150 TB / month data transfer out	$0.080 per GB		

AWS Import/Export

- Unterstützt den Transfer großer Datenmengen in oder aus der Cloud heraus

Available Internet Connection	Theoretical Min. Number of Days to Transfer 1TB at 80% Network Utilization	When to Consider AWS Import/ Export?
T1 (1.544Mbps)	82 days	100GB or more
10Mbps	13 days	600GB or more
T3 (44.736Mbps)	3 days	2TB or more
100Mbps	1 to 2 days	5TB or more
1000Mbps	Less than 1 day	60TB or more

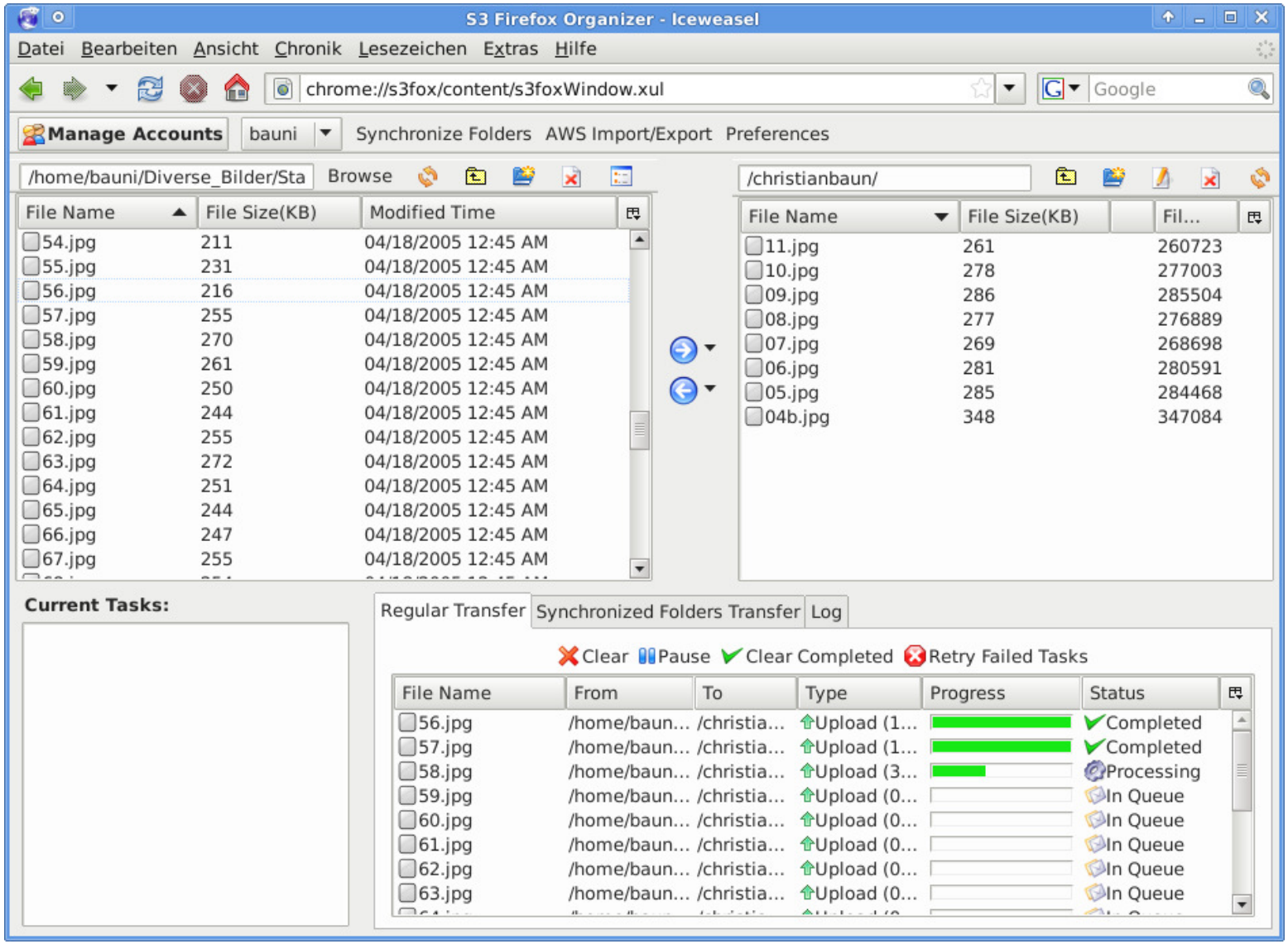
- Ein Datenspeicher (Festplatte) wird an Amazon gesendet und dann vor Ort in einen S3-Bucket kopiert
  - Dateisysteme: NTFS, ext2, ext3 und FAT32 mit maximal 2 TB
- Gebühr pro Datenspeicher: 80 Dollar
- Übertragung vor Ort kostet pro Stunde 2,49 Dollar
- `http://aws.amazon.com/importexport/`

S3 mit s3cmd bedienen

- Für den Zugriff auf Amazon S3 von der Kommandozeile aus, gibt das einfach zu bedienende Werkzeug `s3cmd`
- `http://s3tools.org/s3cmd`

Konfiguration der Zugangsdaten	`s3cmd --configure`
Liste der eigenen Buckets	`s3cmd ls`
Bucket anlegen	`s3cmd mb s3://Bucket`
Objekt hochladen `s3cmd put LokaleDatei s3://Bucket/EntferntesObjekt`	
Inhalt eines Buckets auslesen	`s3cmd ls s3://Bucket`
Objekt herunterladen `s3cmd get s3://Bucket/EntferntesObjekt LokaleDatei`	
Objekt löschen `s3cmd del s3://Bucket/EntferntesObjekt`	
Löschen eines (leeren) Buckets	`s3cmd rb s3://Bucket`

Firefox-Plugin S3Fox: http://www.s3fox.net



Mit der S3 API und boto arbeiten

```
#!/usr/bin/env python

from boto.s3.connection import S3Connection
# Verbindung zu S3 aufbauen
# Variable "conn" zeigt auf ein "S3Connection" Objekt
conn = S3Connection('<aws access key>', '<aws secret key>')

# Bucket "testbucket" erzeugen
conn.create_bucket('testbucket')

# Liste der Buckets erhalten und ausgeben
anfrage_buckets = conn.get_all_buckets()
print(anfrage_buckets)
```

Keys via HTTP POST in S3 hochladen

- Keys in S3 hochladen ist u.a. via HTTP POST möglich
  - `http://doc.s3.amazonaws.com/proposals/post.html`
  - `http://s3.amazonaws.com/doc/s3-example-code/post/post_sample.html`
- Was man zum hochladen benötigt
  - S3-Zugang
    - Access Key und Secret Access Key
  - Bucket
  - Policy-Dokument
  - Signatur
  - HTML-Formular

Keys via HTTP POST in S3 hochladen (2)

```
{ "expiration": "2100-01-01T00:00:00Z",
  "conditions": [
    {"bucket": "<bucket>"},
    ["starts-with", "$acl", ""],
    {"redirect": "<Zieladresse>"},
    ["starts-with", "$key", ""],
    ["starts-with", "$Content-Type", ""]
  ]
}
```

- Für den Namen des Keys kann ein Prefix vorgegeben werden
  z.B. `["starts-with", "$key", "bilder/"],`
- Für den Namen den Content Typs kann ein Prefix vorgegeben werden
  z.B. `["starts-with", "$Content-Type", "image/"],`
- Policy-Dokument wird Base64-kodiert $\Longrightarrow$ **Policy**
- Policy wird mit Secret Access Key verknüpft und dann erneut Base64-kodiert $\Longrightarrow$ **Signatur**

Keys via HTTP POST in S3 hochladen (3)

```
<form action="http://s3.amazonaws.com/<bucket>" method="post"
    enctype="multipart/form-data">
    <input type="hidden" name="key" value="${filename}">
    <input type="hidden" name="acl" value="<ACL>">
    <input type="hidden" name="Content-Type" value="<Content Typ>">
    <input type="hidden" name="redirect" value="<Zieladresse>">
    <input type="hidden" name="AWSAccessKeyId" value="<Access Key>">
    <input type="hidden" name="policy" value="<Policy>">
    <input type="hidden" name="signature" value="<Signatur>">

    <input type="file" name="file">

    <input type="submit" name="submit" value="Upload zu S3">
</form>
```

- Access Control List (ACL) kann sein: `private`, `public-read`, `public-read-write` oder `authenticated-read`
- Werte im Formular müssen mit dem Policy-Dokument übereinstimmen
- Key erfolgreich übertragen $\Longrightarrow$ Redirect des Browsers zur Zieladresse

Einige Anwendungen und Dienste, die S3 nutzen

- Image Hosting Service **SmugMug**
  - Nutzt S3 seit April 2006 um Bilder zu speichern
  - Im April 2008 hat SmugMug angegeben, durch S3 1 Million Dollar gespart zu haben
  - Berechnung: `http://don.blogs.smugmug.com/2006/11/10/amazon-s3-show-me-the-money/`
  - `http://www.smugmug.com`
- Online Backup **Jungle Disk**
  - `http://jungledisk.com`
- Online Backup **ElephantDrive**
  - `http://elephantdrive.com`
- Online Backup **ExEasy NetCDP**
  - `http://www.netcdp.com`
- Online Backup **Dropbox**
  - `http://www.dropbox.com`

Erste NAS-Geräte nutzen schon S3 – Beispiel von HP

- Beispiel: HP MediaSmart Server EX485
- Benutzer kann festlegen welche Daten zum Backup in S3 gespeichert werden sollen

Erste NAS-Geräte nutzen schon S3 – Beispiel von Qnap

Netzwerkspeicher – technische Daten

Modell	TS-239 Pro II	TS-459 Pro
Hersteller/Anbieter	Qnap	Qnap
Web-Adresse	www.qnap.com	www.qnap.com
Hardware und Lieferumfang		
Firmware	3.2.2 (0128T)	3.2.2 (0128T)
Prozessor/RAM	Intel Atom D410 (1,66 GHz)/ 1 GByte DDR2	Intel Atom D510 (1,66 GHz)/ 1 GByte DDR2
LAN-Interface/Link Aggregation/Auto-failover/ Jumbo Frames	2 × Gigabit-Ethernet/✓/✓/✓	2 × Gigabit-Ethernet/✓/✓/✓
Sharing-Funktionen		
FTP/FTP verschlüsselt/abschaltbar	✓/✓/✓	✓/✓/✓
HTTP/HTTPS/abschaltbar	✓/✓/✓	✓/✓/✓
NFS/abschaltbar	✓/✓	✓/✓
AppleShare/abschaltbar	✓/✓	✓/✓
UPnP/abschaltbar	✓/✓	✓/✓
Medienserver per	UPnP-AV (TwonkyMedia), iTunes	UPnP-AV (TwonkyMedia), iTunes
weitere Protokolle	BitTorrent, Bonjour, IPv6, iSCSI, rsync, SNMP, SSH, telnet, WebDAV	BitTorrent, Bonjour, IPv6, iSCSI, rsync, SNMP, SSH, telnet, WebDAV
Printserver/Protokolle	✓/Windows-Share	✓/Windows-Share
Besonderheiten	Unterstützung für DFS u. Amazon S3, IP-Kameras, MySQL, PHP	Unterstützung für DFS u. Amazon S3, IP-Kameras, MySQL, PHP

- *„. . . Daten sichern die Qnap-NAS nicht nur über gängige Mechanismen wie rsync auf andere Server im Netz weg, sondern schicken sie auf Wunsch jetzt auch zeitgesteuert an den Cloud-Speicherdienst Amazon S3. . . “*

Quelle: c't. *Schnelle Gigabit-NAS für zu Hause und das Büro*. 5/2010. S.114

Verfügbarkeit von S3

- Amazon garantiert für S3 eine monatliche Verfügbarkeit von 99.9%

	Downtime (HH:MM:SS)		
Verfügbarkeit	pro Tag	pro Monat	pro Jahr
99.9%	00:01:26	00:43:49	08:45:56

- Wird die Verfügbarkeit unterschritten, erhält man eine Gutschrift
- `http://aws.amazon.com/s3-sla/`

Monthly Uptime Percentage	Service Credit Percentage
Equal to or greater than 99% but less than 99.9%	10%
less than 99%	25%

- Was bringt eine Gutschrift, wenn die eigenen, wichtigen Dienste und Daten nicht mehr erreichbar sind?
- Lösung: Dienste und Daten redundant verfügbar halten!
  - Mehrere Public Cloud Angebote nutzen
  - Private Cloud aufbauen (Hybrid Cloud)

Google Storage

- Speicherdienst für Webobjekte
- `http://code.google.com/apis/storage/`
- Schnittstelle ist kompatibel zu S3
- Funktionalität ist mit S3 identisch
- Für jeden Key und Bucket können die Zugriffsreche festgelegt werden
  - Access Control List (ACL)
  - `http://bucket.commondatastorage.googleapis.com/object`
  - `http://commondatastorage.googleapis.com/bucket/object`
- Zufriff auf Buckets and Keys geschieht via REST
- Kommandozeilenwerkzeug GSutil kann mit Google Storage und S3 arbeiten
  - GSutil basiert auf boto
- Google Storage Manager: SaaS um mit Google Storage zu arbeiten

Nächste Vorlesung

Nächste Vorlesung:

15.4.2011